Federated Kubernetes Clusters Using Amazon EKS and KubeFed

Implementation Guide

Federated Kubernetes Clusters Using Amazon EKS and KubeFed: Implementation Guide

Table of Contents

Federated Kubernetes Clusters Using Amazon EKS and KubeFed

AWS Implementation Guide

AWS Solutions Builder Team

January 2021

This implementation guide describes architectural considerations and configuration steps for deploying a federated Kubernetes environment in Amazon Web Services (AWS) Cloud using Amazon EKS and the open source KubeFed project. It includes links to an AWS CloudFormation template that launches, configures, and runs the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, administrators, DevOps professionals and Platform Engineers who have practical experience architecting in the AWS Cloud and are familiar with Kubernetes technology.

Overview

This solution automates the deployment and federation of two Amazon Elastic Kubernetes Service clusters across multiple AWS Regions using the open source KubeFed project to ensure uninterrupted availability.

You can use the solution's AWS CloudFormation template to deploy the pre-configured bastion host and all necessary dependencies and tools. The solution provides the following capabilities:

- Synchronize your Kubernetes configurations, deployments, and other artifacts between your Amazon EKS clusters to ensure business continuity in case of failure
- Automate the compensation logic of failover events and make your applications resilient to individual cluster failures
- Deploy highly available applications on multiple Amazon EKS clusters in close proximity to end users to reduce latency
- Build architecture compliant with local laws about storing and processing private data

Note
At the date of this solution's publication, the current version of KubeFed is not in *release* state and is only recommended for testing purposes. You must check the status of KubeFed if you plan to use it.

Important
The open source content linked to on these pages is developed, operated, maintained, and owned by third-parties. AWS does not endorse, approve, or certify the content or its sources. The information and links on these pages are provided as-is. AWS makes no warranties and assumes no responsibility or liability for the information, links, and open source content (including licensing information), and any use thereof is at your sole risk. Users should follow their established processes and checks before deploying third-party resources.

Cost

You are responsible for the cost of the AWS services used while running this solution. As of January 2021, the cost for running this solution with default settings in the EU-Central-1 (Frankfurt) and EU-West-1 (Ireland) Regions is approximately **$0.49 per hour**. By default, the solution deploys one working node (m5.large) for each EKS cluster. You can customize configuration of the clusters, including the number and size of instances for the working nodes. Modifying the nodes will impact cost.

Prices are subject to change. For full details, refer to the pricing webpage for each AWS service used in this solution.

AWS Services	Description	Cost per hour
2 x EC2 m5.large	One working node (~$0.107 per hour) for two EKS clusters in different Regions	$0.222
2 x Amazon EKS	Each EKS cluster $0.10	$0.20

AWS Services	Description	Cost per hour
2 x VPC and NAT Gateway	Price per NAT gateway $0.052 per hour in EU-Central-1 (Frankfurt) and $0.048 in EU-West-1 (Ireland)	$0.10
EC2 t3.micro (bastion)	Price per hour	$0.012
Total		**~ $0.534 per hour**

Architecture overview

Deploying this solution with the default parameters builds the following environment in the AWS Cloud.

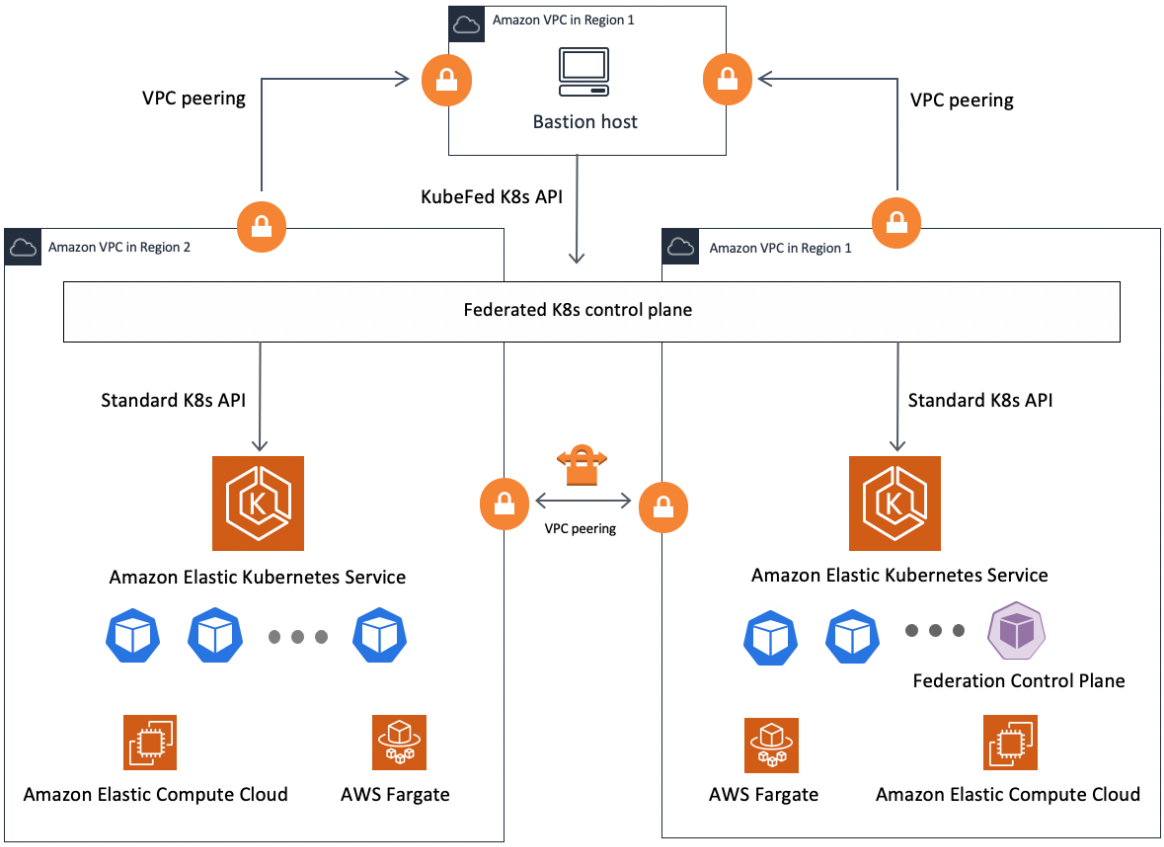

Figure 1: Federated Kubernetes Clusters Using Amazon EKS and KubeFed solution architecture

The AWS CloudFormation template first deploys a bastion host in a new Amazon VPC and provisions the following necessary resources:

- An Amazon VPC
- Public and private subnets
- A NAT gateway
- An internet gateway

- Amazon EC2 for the bastion host

After successfully deploying the bastion host, the **eksfedctl** tool automatically deploys the following items:

- Two Amazon VPCs for Amazon EKS clusters in your selected AWS Regions
- Two Amazon EKS clusters in different AWS Regions, each in its own Amazon VPC
- Amazon VPC peering between three Amazon VPCs for secure communication between the bastion host and the federated Amazon EKS clusters
- A federation control panel that serves as a proxy between the Kubernetes administrator and the deployed Amazon EKS clusters based on the open source KubeFed project

The bastion host is the single point of administration for Amazon EKS, Kubernetes resources, and this solution's deployment and configuration.

Design considerations

KubeFed

This solution automates the deployment of multiple well-architected Amazon EKS clusters across AWS Regions and federates them using the official Kubernetes open-source project - KubeFed. Review the following KubeFed considerations before deploying the solution.

- At the date of this solution's publication, the current version of KubeFed is not in *release* state and is only recommended for testing purposes. You must check the status of KubeFed if you plan to use it.
- This solution provisions only two Amazon EKS clusters. However, you can extend this solution by using the KubeFed join command to add clusters. This solution does not automate the extension.
- KubeFed provides an additional management layer on top of your Kubernetes clusters and can help manage complex applications that stretched across deployments, but you should not consider it to be the unified management point for all clusters. KubeFed only supports scenarios for federation of resources and does not cover all Kubernetes functionality.
- KubeFed does not have mechanics and capabilities to sync persistent storage between clusters.
- KubeFed does not support all Kubernetes entities in federation mode. For more information, refer to the Kubernetes Cluster Federation GitHub.

> **Note**
> Refer to Running highly available applications for best practices for deploying applications on Amazon EKS.

We recommend two options for deployment:

- Scope your deployment YAML files to a small subset of clusters before broadcasting changes to all federated clusters. You can use cluster selectors or unique namespaces. The following code is an example deployment YAML file. More examples are provided with the solution in the following location: `~/.eksfed/examples/federated-apps/01-simple-app-example/`

```
spec:
  placement:
    clusters:
      - name: cluster2
      - name: cluster1
```

- Test your changes in a staging/testing federation environment before rolling them out to your main federated clusters.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model reduces your operational burden because AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. For more information about security on AWS, visit AWS Cloud Security.

Access to Amazon EKS clusters is only configured for private AWS networks and is not accessible from public internet. You can customize the configuration using the Amazon EKS console, or using **eksctl**. For more information, refer to Amazon EKS endpoint access control.

General security approach

This solution minimizes its components network exposure to the public internet. A bastion host is provisioned in a private subnet without a SSH key, so you can only access it through AWS Systems Manager Session Manager. Amazon Elastic Kubernetes Service (Amazon EKS) clusters are provisioned only with private endpoints and are able to communicate with each other only through VPC peering for the private subnets.

We recommend that you do not enable public endpoints for Amazon EKS and do not to move the bastion host instance to a public subnet, or add an SSH key.

Amazon EKS Role-based access control (RBAC) and AWS Identity and Access Management (IAM)

By default, this solution's `FederationBootstrapperRole` has full access to the provisioned clusters.

During solution deployment, two new IAM groups are created: `ClusterAdminGroup` and `ClusterUserGroup`. Users in these IAM groups can assume the following correlated roles: `ClusterAdminRole` and `ClusterUserRole`. These roles are mapped to corresponding EKS Role-based access controls (RBAC): `system:masters` and `system:authenticated`. RBAC allows the IAM roles to have admin and user rights respectively.

For more information, refer to Managing users or IAM roles for your cluster in the *Amazon EKS User Guide*.

Security groups

This solution utilizes security groups to control and isolate traffic between the bastion host and EKS clusters. The security groups within each cluster restrict the communication for the internal cluster components and guarantee that they are not accessible from outside of each cluster.

We recommend that you do not to change the default EKS security groups. We also recommend that you review the bastion host security group and restrict its access if necessary.

AWS CloudFormation template

This solution uses AWS CloudFormation to automate deployment in the AWS Cloud. It includes the following CloudFormation template, which you can download before deployment.

federated-amazon-eks-clusters-on-aws.template: Use this template to launch the bastion host that provisions the solution and all associated components. The default configuration deploys the bastion host with a t3.micro instance size (based on Amazon Linux2 OS) using the latest available Amazon Machine Image (AMI) according to the chosen Region. You can customize the template and its parameters based on your specific needs.

Note
AWS CloudFormation resources are created from AWS Cloud Development Kit (AWS CDK) constructs.

Automated deployment

Before you launch the solution, review the architecture, configuration, and other considerations discussed in this guide.

Use the instructions in this section to configure and deploy the Federated Kubernetes Clusters Using Amazon EKS and KubeFed solution into your account.

Time to deploy: Approximately 60 minutes

Prerequisites

You must have an AWS account and an IAM user with either Administrator permissions or least permissions. These permissions are required to deploy the solution and the corresponding components in AWS infrastructure.

Deployment overview

The deployment process consists of two stages:

- The first stage is provisioning a new VPC with a bastion host that is the single point of management for your EKS clusters, with access to all installed dependencies and libraries. The bastion host has connectivity to two Amazon VPCs.
- The second stage allows you to customize your EKS deployment configuration, and use the **eksfedctl** tool to provision another two Amazon VPCs and two private EKS clusters without public endpoints. You can also and set up federation between the clusters.

Amazon EKS clusters do not have public endpoints. Therefore, you must establish a place from which you will have connectivity to your EKS clusters. This solution uses the bastion host to simplify this connectivity.

Step 1. Attaching the security policy to a user

To launch the stack, the AWS account administrator must attach the provided policy to the role or the user who will perform the federation configuration on the bastion host.

> **Note**
> We recommend that you attach the security policy to an AWS IAM role, and then attach that role to the user.

Use the following process to attach the solution's security policy to the solution's user.

1. Use the following link to download the security policy: solution-user-policy.json.
2. Create a new user policy.
3. Attach the created policy to the IAM user or IAM role.

For more information, refer to Adding IAM identity permissions in the *AWS IAM User Guide*.

Step 2. Launch the stack

This automated AWS CloudFormation template deploys Federated Kubernetes Clusters Using Amazon EKS and in the AWS Cloud. You must have either administrator access or a user account with the appropriate security policy (Step 1) before launching the stack.

Note
You are responsible for the cost of the AWS services used while running this solution. Refer to the Cost (p. 2) section for more details. For full details, refer to the pricing webpage for each AWS service used in this solution.

1. Sign in to the AWS Management Console and select the button below to launch the `federated-amazon-eks-clusters-on-aws` AWS CloudFormation template.

You can also download the template as a starting point for your own implementation.

2. The template launches in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar.

Note
This solution uses the Amazon EKS service, which is currently available in specific AWS Regions only. Therefore, you must launch this solution in an AWS Region where Amazon EKS is available. For the most current availability by Region, refer to the AWS Regional Services List.

3. On the **Create stack** page, verify that the correct template URL shows in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack.
5. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
Bastion host type	`t3.micro`	Amazon EC2 instance type for the bastion host.
Bastion host AMI	`/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2`	AMI to be used on the bastion host. This parameter is auto-resolved from the SSM ParameterStore for AMI Linux 2 AMI (latest).

6. Choose **Next.**
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a **CREATE_COMPLETE** status in approximately five minutes.

Step 3. Provision federated Amazon EKS infrastructure and verify the results

Note
You can use an existing machine to manage federation and Amazon EKS clusters using scripts, but you must first establish connectivity to a newly created Amazon EKS cluster.

Provision the Federated Amazon EKS infrastructure

1. Sign in to the Amazon EC2 console in the Region where you deployed the solution's stack.
2. Search for the Amazon EC2 instance with name similar to **bastion-*{stack-name}***.
3. Select the instance and choose **Connect**.
4. Select **Session Manager** and choose **Connect**.
5. In the new browser window, the terminal window opens to **sh-4.2$** in the console.
6. Enter tmux in the console to start a new multiplexing session.

   This step is required to preserve the scripts being run.

   Under the *tmux* session, the scripts will continue to run even if you are disconnected. If you are disconnected, you can reconnect to the bastion host using SSM and the `tmux attach` command.
7. Enter the following command, replacing the *{Region1}* and *{Region2}* placeholders with the Region names of your choice:

```
eksfedctl create [-h] [-f FILE] [-n NAME] [-r REGION1 REGION2]
```

Example: `eksfedctl create --regions eu-west-1 eu-central-1`

Arguments	Default	Description
-r --regions	*<Requires input>*	Set the Regions where the EKS clusters will be deployed.
-f --file	<Optional input>	The configuration file for federation in yaml format. An example is provided in the repository.
-n --name	<Optional input>	The prefix name for EKS clusters. If you do not provide a name, the prefix is randomized.

After approximately 60 minutes, all resources and components (including VPC peering to your bastion host from the new Amazon EKS cluster VPC) are created and you can verify federation.

Note
By default, VPC logs are only active for the bastion's VPC.

(Optional) Verify the federation and deploy the test application

1. After deployment, run the following command to verify federation on the bastion host:

```
kubectl -n kube-federation-system get kubefedclusters
```

The code will return something similar to the following example (NAME, and AGE values will be different in your output):

```
NAME              AGE   READY
federated-eks-1   22h   True
federated-eks-2   22h   True
```

2. Deploy the example federated application (nginx) using `install.sh` from the folder on the bastion host:

```
cd ~/.eksfed/examples/federated-apps/01-simple-app-example/

./install.sh
```

The sample YAML files for the example application can be modified to meet your use case. You can use `kubectl` to modify the following files:

1. `Namespace.yaml #` contains *test* namespace definition.
2. `federated-namespace.yaml #` contains ***FederatedNamespace*** definition based on *test* namespace.
3. `federated-nginx.yaml #` contains ***FederatedDeployment*** definition in *test* namespace based on *nginx* image.
4. `federated-deployment-rsp.yaml #` contains ***ReplicaSchedulingPreference*** definition to distribute replicas among clusters for ***FederatedDeployment*** in *test* namespace.

   Successful installation deploys the `nginx` application in federated mode. To verify installation, enter the command: `kubectl get pods -n test`.

   The command returns the following code:

```
NAME                               READY   STATUS    RESTARTS   AGE
test-deployment-7d5848c88c-xxxxx   1/1     Running   0          21h
test-deployment-7d5848c88c-xxxxx   1/1     Running   0          21h
test-deployment-7d5848c88c-xxxxx   1/1     Running   0          21h
test-deployment-7d5848c88c-xxxxx   1/1     Running   0          21h
test-deployment-7d5848c88c-xxxxx   1/1     Running   0          21h
```

This command returns the pods in the current cluster only. To view the federated portion in the second cluster, run the following commands:

a. `kubectl config get-contexts`

   The command returns the following, where *eksfed* is your stack name:

```
CURRENT   NAME          CLUSTER      AUTHINFO     NAMESPACE
*         eksfed-1      eksfed-1     eksfed-1
          eksfed-2      eksfed-2     eksfed-2
```

b. `kubectl get pods -n test --context {context NAME}`

   The command returns another five test deployments from the second Amazon EKS cluster in federation.

(Optional) Customize Amazon EKS deployment

Use the EksFedctl config file to customize your EKS cluster configuration, including number and types of nodes within managed and unmanaged nodegroups, CloudWatch settings, and setup AWS Fargate profiles.

Example: `eksfedctl create -f CONFIG_FILE.yaml`

Refer to the Federated Amazon EKS Clusters on AWS GitHub to copy sample configuration files that you can use or customize to fit your needs:

- 01-simple-cluster.yaml—Create federated EKS clusters with EKS managed nodegroups
- 02-nodegroup.yaml—Create federated EKS clusters with unmanaged nodegroups
- 03-fargate-profile.yaml—Create federated EKS clusters with AWS Fargate Profiles

EksFedctl config inherits the structure from **EksCtl** config file with some limitations. The file has the following format:

```
apiVersion: fedk8s/v1
kind: FederatedEKSConfig

metadata:
  name: CLUSTER_NAME
  regions:
    - REGION_1
    - REGION_2
spec:
    iam:
       ...
    nodeGroups:
       ...
    managedNodeGroups:
      ...
    fargateProfiles:
     ...
    git:
      ...
    cloudWatch:
      ...
```

The following EksCtl config sections are accepted: IAM, nodeGroups, managedNodeGroups, fargateProfiles, GIT, and CloudWatch. Other root elements for the EksCtl config file are not supported and are ignored.

Uninstall the solution

You can use the included **eksfedctl** tool to automatically uninstall all of the solution's resources. The tool uses a preconfigured .env file (stored in the `/home/ssm-user` folder) that includes all of the necessary parameters for automatic uninstallation.

```
eksfedctl destroy -f {full path to .env file} (e.g. ~/{stack
name}.env or /home/ssm-user/{stack name}.env)
```

Example:

```
eksfedctl destroy -f ~/eks-fed-10.env
```

The **eksfedctl** tool automatically performs the following steps to clean up the resources:

1. Remove all peering connections that were created in all Regions with EKS clusters and bastion host.
2. Remove all NodeGroups from both EKS clusters.
3. Remove EKS clusters in both Regions.

After running the **eksfedctl** `destroy` command, remove the host stack:

1. Launch the AWS CloudFormation console in the Region with the bastion host.
2. Select your stack and choose **Delete**.
3. Select **Delete stack**.

Additional resources

AWS services

- AWS CloudFormation
- Amazon Elastic Kubernetes Service (Amazon EKS)
- AWS Cloud Development Kit (AWS CDK)

Other links

- Kubernetes KubeFed

Collection of operational metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When enabled, the following information is collected and sent to AWS:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each Federated Kubernetes Clusters Using Amazon EKS deployment
- **Timestamp:** Data-collection timestamp
- **Instance Data:** Count of the state and type of instances that are managed by the Amazon EC2 Scheduler in each AWS Region

Example data:

```
Running: {t2.micro: 2}, {m3.large:2}
Stopped: {t2.large: 1}, {m3.xlarge:3}
```

AWS owns the data gathered through this survey. Data collection is subject to the AWS Privacy Policy. To opt out of this feature, modify the AWS CloudFormation template mapping section from:

```
"Send" : {
   "AnonymousUsage" : { "Data" : "Yes" }
},
```

to:

```
"Send" : {
   "AnonymousUsage" : { "Data" : "No" }
},
```

Source code

Visit the GitHub repository to download the templates and scripts for this solution, and to share your customizations with others. The Federated Kubernetes Clusters Using Amazon EKS and KubeFed template is generated using the AWS Cloud Development Kit (AWS CDK). Refer to the README.md file for additional information.

Contributors

The following individuals contributed to this document:

- Viktor Kiselev
- Roman Boiko
- Sergey Pugachev
- Aleksandr Iziumov
- Sergey Kurson

Revisions

Date	Change
January 2021	Initial release
January 2021	Solution name changed from *Federated Amazon EKS Clusters on AWS* to *Federated Kubernetes Clusters Using Amazon EKS and KubeFed*

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents AWS current product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. AWS responsibilities and liabilities to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.